CLAIMS

We claim:

1. A method for adding functionality in order to access information, comprising:

accessing existing object code, said existing object code includes a first method, said first method is capable of providing a result; and

adding new code to said first method, said new code provides said result to said additional method.

2. A method according to claim 1, wherein:

said result includes a data item to be returned by said first method

3. A method according to claim 1, wherein:

said result includes a reference to an exception.

4. A method according to claim 1, wherein said step of adding new code includes:

adding code that stores said result for said first method from an operand stack;

adding code that prepares said operand stack for an invocation of said additional method;

adding code that invokes said additional method, including providing said result to said additional method; and

adding code that resets said operand stack with respect to said result to a state existing prior to storing said result.

5. A method according to claim 4, wherein said step of adding new code further includes:

adding code that returns said result after resetting said operand stack, said result is a return value.

6. A method according to claim 4, wherein:

said result includes an exception; and

said step of adding new code further includes adding code that throws said exception after said step of resetting, said result represents an exception.

7. A method according to claim 4, wherein said step of adding new code further includes:

adding code that jumps to a subroutine representing a Finally block after invoking said additional method; and

adding code that is to be executed after returning from said subroutine.

8. A method according to claim 1, wherein:

said first method is a Java method.

9. A method according to claim 1, wherein said step of adding new code includes:

adding start byte code;

adjusting byte code indices;

adding exit byte code; and

modifying an exception table for said first method.

10. A method according to claim 9, wherein said step of adding exit byte code

includes:

adding byte code to report said result and jump to a subroutine representing a Finally block;

adding byte code to report an exception and jump to said subroutine representing said Finally block; and

adding byte code for said subroutine representing said Finally block.

11. A method according to claim 1, wherein said step of adding new code includes:

adding Try-Finally functionality.

12. A method for accessing information, comprising:

storing a result for a first method from an operand stack;

preparing said operand stack for an invocation of a second method;

invoking said second method, including providing said result to said second method; and

resetting said operand stack with respect to said result to a state existing prior to said step of storing said result.

13. A method according to claim 12, wherein:

said result includes a data item to be returned by said first method

14. A method according to claim 13, further comprising:

returning said result after said step of resetting.

15. A method according to claim 12, wherein:

said result includes a reference to an exception.

16. A method according to claim 15, further comprising:

throwing said exception after said step of resetting.

17. A method according to claim 12, further comprising:

performing said second method in response to said step of invoking.

18. A method according to claim 12, further comprising:

performing one or more instructions of said first method prior to said step of storing said result, said step of performing one or more instructions includes generating said result.

19. A method according to claim 12, further comprising:

returning said result subsequent to said step of resetting;

jumping to a subroutine representing a Finally block after invoking said second method and prior to returning said result; and

returning from said subroutine prior to returning said result.

20. A method according to claim 12, wherein:

said first method is a Java method.

21. A method according to claim 12, further comprising:

modifying byte code for said first method to add new code that performs said steps of storing, preparing, invoking and resetting.

22. A method according to claim 21, wherein said step of modifying includes:

adding start byte code;

adjusting byte code indices;

adding exit byte code; and

modifying an exception table for said first method.

23. A method according to claim 22, wherein said step of adding exit byte code includes:

adding byte code to report said result and jump to a subroutine representing a Finally block;

adding byte code to report an exception and jump to said subroutine representing said Finally block; and

adding byte code for said subroutine representing said Finally block.

24. A method according to claim 21, wherein said step of modifying includes:

adding Try-Finally functionality.

25. A method according to claim 12, further comprising:

performing said second method, including accessing said result.

26. A method according to claim 12, further comprising:

performing said second method, including storing said result for use outside of a thread that includes said first method.

27. A method according to claim 12, further comprising:

performing said second method in response to said step of invoking, said second method stores said result;

performing one or more instructions of said first method prior to said step of storing said result, said step of performing one or more instructions includes generating

said result, said first method is a Java method; and

modifying byte code for said first method to add new code that performs said steps of storing, preparing, invoking and resetting.

28. A method according to claim 27, further comprising:

returning said result;

jumping to a subroutine representing a Finally block after invoking said second method and prior to returning said result; and

returning from said subroutine prior to returning said result.

29. One or more processor readable storage devices having processor readable code embodied on said processor readable storage devices, said processor readable code for programming one or more processors to perform a method comprising:

accessing existing object code, said existing object code includes a first method, said first method is capable of providing a result; and

adding new code to said first method, said new code provides said result to said additional method.

30. One or more processor readable storage devices according to claim 29, wherein:

said result is a data item to be returned by said first method

31. One or more processor readable storage devices according to claim 29, wherein:

said result is a reference to an exception.

32. One or more processor readable storage devices according to claim 29,

wherein said step of adding new code includes:

adding code that stores said result for said first method from an operand stack;

adding code that prepares said operand stack for an invocation of said additional method;

adding code that invokes said additional method, including providing said result to said additional method; and

adding code that resets said operand stack with respect to said result to a state existing prior to storing said result.

33. One or more processor readable storage devices according to claim 29, wherein:

said first method is a Java method.

34. One or more processor readable storage devices according to claim 29, wherein said step of adding new code includes:

adding start byte code;

adjusting byte code indices;

adding exit byte code; and

modifying an exception table for said first method.

35. One or more processor readable storage devices according to claim 34, wherein said step of adding exit byte code includes:

adding byte code to report said result and jump to a subroutine representing a Finally block;

adding byte code to report an exception and jump to said subroutine representing said Finally block; and

adding byte code for said subroutine representing said Finally block.

36. An apparatus that adds functionality in order to access information, comprising:

a communication interface;

a processor readable storage device; and

one or more processors in communication with said processor readable storage device and said communication interface, said one or more processors perform a method comprising:

access existing object code, said existing object code includes a first method, said first method is capable of providing a result, and

adding new code to said first method, said new code provides said result value to said additional method.

37. An apparatus according to claim 36, wherein:

said result is a data item or a reference to an exception.

38. An apparatus according to claim 36, wherein said step of adding new code includes:

adding code that stores said result for said first method from an operand stack;

adding code that prepares said operand stack for an invocation of said additional method;

adding code that invokes said additional method, including providing said result to said additional method; and

adding code that resets said operand stack with respect to said result to a state existing prior to storing said result.

39. An apparatus according to claim 36, wherein said step of adding new code

includes:

adding start Java byte code;

adjusting Java byte code indices;

adding exit Java byte code; and

modifying an exception table for said first method.

40. An apparatus according to claim 39, wherein said step of adding exit byte code includes:

adding byte code to report said result and jump to a subroutine representing a Finally block;

adding byte code to report an exception and jump to said subroutine representing said Finally block; and

adding byte code for said subroutine representing said Finally block.

41. An apparatus that adds functionality to existing code in order to access information, comprising:

a communication interface;

a processor readable storage device; and

one or more processors in communication with said processor readable storage device and said communication interface, said one or more processors perform a method comprising:

storing a result for a first method from an operand stack,

preparing said operand stack for an invocation of a second method,

invoking said second method, including providing said result to said second method, and

resetting said operand stack with respect to said result to a state existing prior to said step of storing said result.

42. An apparatus according to claim 41, wherein:

said result is a data item to be returned by said first method

43. An apparatus according to claim 41, wherein:

said result is a reference to an exception.

44. An apparatus according to claim 41, wherein:

said first method is a Java method.

45. An apparatus according to claim 41, wherein said method further comprises:

modifying byte code for said first method to add new code that performs said steps of storing, preparing, invoking and resetting.

46. An apparatus according to claim 45, wherein said step of modifying includes the steps of:

adding start byte code;

adjusting byte code indices;

adding exit byte code; and

modifying an exception table for said first method.

47. An apparatus according to claim 46, wherein said step of adding exit byte code includes:

adding byte code to report said result and jump to a subroutine representing a Finally block;

adding byte code to report an exception and jump to said subroutine representing

said Finally block; and

adding byte code for said subroutine representing said Finally block.

5